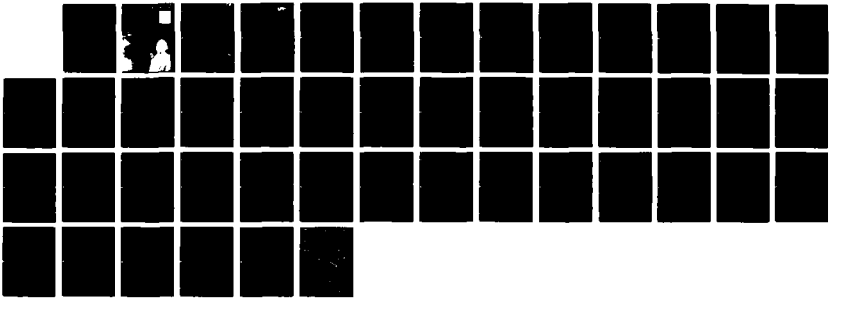
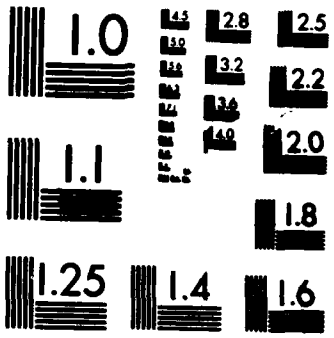
AD-A182 943 MARINE CORPS COMBAT READINESS EVALUATION SYSTEM SOFTWARE APPLICATIONS (MC..(U) GEORGE WASHINGTON UNIV WASHINGTON DC INST FOR MANAGEMENT SCIE.. W E CAVES 1/1

UNCLASSIFIED 30 AUG 85 GWU/IMSE/SERIAL-T-502/85 F/G 12/5 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DTIC FILE COPY

AD-A182 943

Marine Corps Combat Readiness Evaluation System Software Applications (MCCRESSA) Support Programs

by

W. E. Caves

THE GEORGE WASHINGTON UNIVERSITY

STUDENTS FACULTY STUDY R
ESEARCH DEVELOPMENT FUT
URE CAREER CREATIVITY CO
MMUNITY LEADERSHIP TECH
NOLOGY FRONTIE ESIGN
ENGINEERING APP ENC
GEORGE WASHIN NIV

DTIC ELECTE JUN 08 1987 S D E

This document has been approved for public release and sale; its distribution is unlimited.

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

Marine Corps Combat Readiness Evaluation System Software Applications (MCCRESSA) Support Programs

by

W. E. Caves

Readiness Research
GWU/IMSE/Serial T-502/85
August 1985

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Washington, DC 20052

Institute for Management Science and Engineering

Research Supported
by
Contract N00014-85-C-0716
Project NR 347 131
Office of Naval Research

DTIC
ELECTE
JUN 0 8 1987
S D
E

This document has been approved for public sale and release; its distribution is unlimited.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER GWU/IMSE/Serial T-502/85	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD-A182 943
4. TITLE (and Subtitle) MARINE CORPS COMBAT READINESS EVALUATION SYSTEM SOFTWARE APPLICATIONS (MCCRESSA) SUPPORT PROGRAMS		5. TYPE OF REPORT & PERIOD COVERED SCIENTIFIC 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) W. E. CAVES		8. CONTRACT OR GRANT NUMBER(s) Supported by N00014-85-C-0716
9. PERFORMING ORGANIZATION NAME AND ADDRESS GEORGE WASHINGTON UNIVERSITY INSTITUTE FOR MANAGEMENT SCIENCE AND ENGINEERING WASHINGTON, D.C. 20052		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS OFFICE OF NAVAL RESEARCH CODE 411 S&P ARLINGTON, VA 22217		12. REPORT DATE 30 August 1985 13. NUMBER OF PAGES 39
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC SALE AND RELEASE; DISTRIBUTION UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

COMBAT READINESS
READINESS EVALUATION
READINESS
TRAINING PROFICIENCY
PERFORMANCE ASSESSMENT

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A set of five support programs for the Marine Corps Combat Readiness Evaluation System Software Applications (MCCRESSA) are documented. These support programs perform common operations for many MCCRESSA functional program which load them as needed. The support programs documented here are Program Control (PGMCTL), Set Program Parameters (SETPRM), Get MCCRES Master File Header Information (MFINFO), Get MCCRES Requirement Counts File Header Information (CRINFO), and Find Data Set (FINDDS).

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Washington, DC 20052

Institute for Management Science and Engineering

Marine Corps Combat Readiness Evaluation System Software Applications (MCCRESSA) Support Programs

by

W. E. Caves

Abstract
of
Readiness Research
GWU/IMSE/Serial T-502/85
30 August 1985

A set of five support programs for the Marine Corps Combat Readiness Evaluation System Software Applications (MCCRESSA) are documented. These support programs perform common operations for many MCCRESSA functional programs which load them as needed. The support programs documented here are Program Control (PGMCTL), Set Program Parameters (SETPRM), Get MCCRES Master File Header Information (MFINFO), Get MCCRES Requirement Counts File Header Information (CRINFO), and Find Data Set (FINDDS).

QUALITY INSPECTED 4

Research Supported
by
Contract N00014-85-C-0716
Project NR 347 131
Office of Naval Research

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

TABLE OF CONTENTS

1.	Introduction	1
2.	Program Control (PGMCTL)	2
2.1	PGMCTL Parameter	2
2.2	Control File	5
2.3	Message Templates	6
2.4	Program Control Value Records	7
2.5	PGMCTL Error Messages	7
3.	Set Program Parameters (SETPRM)	8
3.1	Program Parameters	8
3.2	Control File	13
3.3	Commands	15
3.4	Edit Control	18
3.5	Justification Control	19
3.6	Relational Expressions	20
3.7	Program Messages	21
4.	Get MCCRES Master File Header information (MFINFO)	27
4.1	Master File Header Parameter	27
4.2	Program Messages	31
5.	Get MCCRES Requirement Counts File Header Information (CRINFO)	33
5.1	Requirement Counts File Header Parameter	33
5.2	Program Messages	35
6.	Find Data Set (FINDDS)	36
6.1	Operation	36
6.2	Error Messages	37

LIST OF FIGURES

2.1	PGMCTL Major Logical Flow	3
3.1	STDPARM Copy Code	9
3.2	ROOTPARM.EDLLIB	10
3.3	ROOTEQU.EDLLIB	10
3.4	Keyword Table Definition	11
3.5	ROOTKWRD.EDLLIB	12
3.6	ROOTEND.EDLLIB	12
3.7	Code to set Number of Keyword Entries	13
4.1	MFINFO MCCRES Header Information Parameter, EDL Definition	28
4.2	MFINFO MCCRES Header Information Parameter, COBOL Definition	29
4.3	Header Detail Definition	30
5.1	Requirement Counts File Header Information Parameter, EDL	34
5.2	Requirement Counts File Header Information Parameter, COBOL	34
6.1	FINDDS Indirect Parameter Area Map	36

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Washington, DC 20052

Institute for Management Science and Engineering

Marine Corps Combat Readiness Evaluation System Software Applications (MCCRESSA) Support Programs

by

W. E. Caves

Readiness Research
GWU/IMSE/Serial T-502/85
30 August 1985

1. Introduction

The Marine Corps Combat Readiness Evaluation System (MCCRES) has been operational since July 1978 and, since that time, the Readiness Research group has performed research on MCCRES concepts, procedures, and data. Since the third quarter of Fiscal 1981, this research has been supported by an IBM Series/1 installed at The George Washington University. This support has allowed the development of MCCRES Software Applications (MCCRESSA) programs in a COBOL environment similar to that existing within Headquarters USMC (HQMC). Thus, the programs developed to support the MCCRES research conducted at The George Washington University are also suitable for operation at HQMC and, in some instances with modification, at Marine Units in the field. In fact, such MCCRESSA programs are presently operating at HQMC and in the field.

This paper describes a set of five programs which are loaded by MCCRESSA functional programs to perform common operations. The support programs are Program Control (PGMCTL), Set Program Parameters (SETPRM), Get MCCRES Master File Header Information (MFINFO), Get MCCRES Requirement Counts File Header Information (CRINFO), and Find Data Set (FINDDS), which are documented in Sections 2 through 6, respectively.

References [1] through [7] and [9] through [11] contain related MCCRES material. Reference [8] is a relevant IBM publication.

2. Program Control (PGMCTL)

Many MCCRESSA functional programs, written or rewritten after June 1984, have some of their housekeeping, end-of-job and terminal error message handling routines collected into a common overlay program (PGMCTL). This program is typically loaded first and last by a MCCRESSA program root. It allows the tailoring of the housekeeping program logo display, selected program control constants, terminal error message text, and the end-of-job display; all without recompiling any program.

PGMCTL requires a parameter passed from the MCCRESSA functional program root that loaded it and a control file which must be resident on the same EDX volume from which PGMCTL was loaded. The operation of PGMCTL is transparent to the user who need not know of its existence. The requisite control file, like the requisite parameter, is the responsibility of the MCCRESSA programmer. Each are described below. Figure 2.1, PGMCTL Major Logic Flow, augments the discussion to follow.

2.1. PGMCTL Parameter

PGMCTL requires a 24 byte parameter which must begin on a word boundary. Its format is as follows.

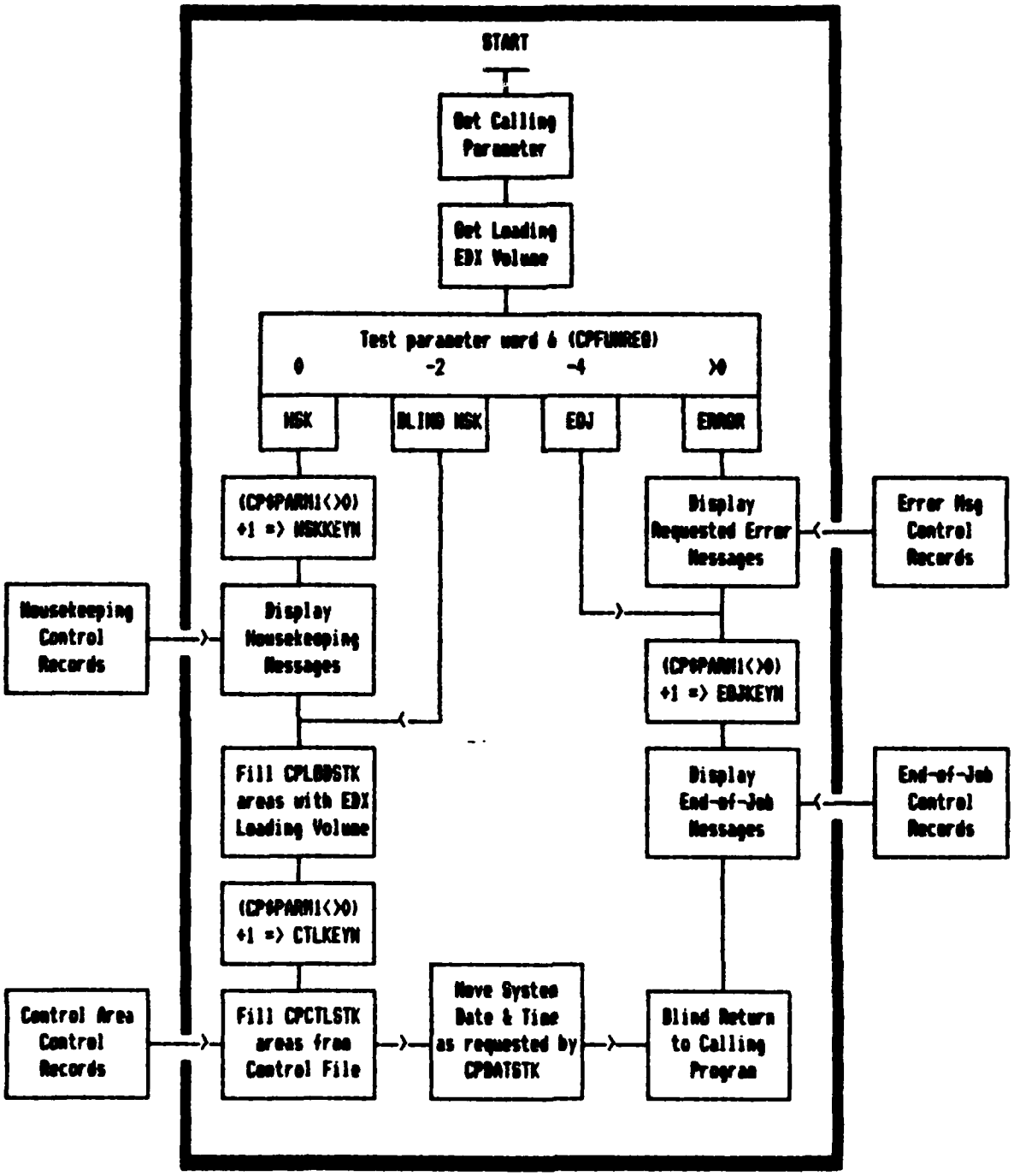
CPCTLFN Bytes 1 thru 8: The data set name of the loading program's control file.

CP$PARM1 Bytes 9 and 10: The value of the $PARM1 word of the loading program. This allows PGMCTL to determine if the loading program was loaded by $L or by $JOBUTIL. Houskeeping messages displayed, the contents of the program control areas filled, and end-of-job displays may be conditional upon this value.

CPERRNO or
CPFUNREQ Bytes 11 and 12: This word indicates the control function be be performed. When greater than zero, the value is taken to be the number of a terminal error. Otherwise, the value indicates the control function as follows:

```
 0 -> Housekeeping Call
-2 -> Blind Houskeeping Call
-4 -> Normal End-of-Job Call
```

CPDATSTK Bytes 13 and 14: This word locates a stack of two addresses in the loading program which is processed during either type of housekeeping call. The first address locates an eight byte area which is to receive the system date in the form of mm/dd/yy. If the loading program does not request the system date, this address must be zero. The second address locates an eight byte area which is to receive the system time in the form of hh:mm:ss. If the loading program does not request the system time, this address must be zero. If



Note: All Control Records are read from the Program Control File.

PGMCTL Major Logical Flow

Figure 2.1

neither the system date nor the system time are requested, CPDATSTK itself may be zero.

CPHSKSTK Bytes 15 and 16: This word locates a stack of words used in constructing housekeeping messages (see Section 2.3 below). Housekeeping message templates are contained in the program control file. If these do not require any program information, CPHSKSTK may be zero. Typically, CPHSKSTK locates a three word stack where the first word locates an eight character area containing the program name, the second word's value is the program's version number, and the third word's value is the program's modification level within the current version. The actual stack required is determined by the housekeeping message templates included in the program's control file but must at least begin with the above three words. The control file key for selecting housekeeping message records is HSK00n where n is set to one or two depending upon the manner in which the MCCRESSA functional program was loaded, i.e., $L or $JOBUTIL, respectively.

CPCTLSTK Bytes 17 and 18: This word locates a stack of word pairs, each of which defines a program control area which is to be filled from the control file during a housekeeping call. The control file key for selecting records is CTL00n where n is set to one or two depending upon the manner in thich the MCCRESSA functional program was loaded, i.e., $L or $JOBUTIL, respectively. There must be exactly one selected control record for each word pair in the stack. The first word of a word pair gives the length of the control area located by the second word. The stack is terminated by a length of zero.

CPEOJSTK Bytes 19 and 20: This word locates a stack of words used in constructing end-of-job messages (see Section 2.3 below). End-of-job message templates are contained in the program control file. If these do not require any program information, CPEOJSTK may be zero. The actual stack required is determined by the end-of-job message templates included in the program's control file. The control file key for selecting end-of-job message records is EOJ00n where n is set to one or two depending upon the manner in which the MCCRESSA functional program was loaded, i.e., $L or $JOBUTIL, respectively.

CPLODSTK Bytes 21 and 22: This word locates a stack of words that each locate a six byte program area that must be set during a housekeeping call to the EDX volume from which PGMCTL was loaded. The stack ends with the value zero.

CPERRSTK Bytes 23 and 24: This word locates a stack of words used in constructing terminal error messages (see Section 2.3 below). Terminal error message templates are contained in the program control file. If these do not require any program information, CPERRSTK may be zero. The actual stack required is determined by the terminal error message templates included in the program's control file. The control file key for selecting terminal error message templates is ERRnnn where nnn is the value of CPERRNO. When a terminal error call is made, at least one terminal error message template must match the terminal error number. If this is not the case, a message beginning with the program name located by the first word of CPHSKSTK followed by ': ERROR ' and the error number is displayed.

2.2. Control File

Each MCCRESSA functional program which calls PGMCTL must also supply the name of a control file as the first eight bytes of the PGMCTL parameter. This control file is usually prepared by a typical editor. As such, 128 byte logical records are packed two to the 256 byte EDX record and only the first 72 positions of a logical record are used as a control record. Finally, each control record is divided into three fields as follows.

Position	Use
1-3	Control record type code.
4-6	Control record number.
8-72	Control record body.

Five types of control records are allowed. These are: Comment Records, blank type code - not processed; Housekeeping Message Templates, HSK type code; Program Control Value Records, CTL type code; Terminal Error Message Templates, ERR type code; and End-of-Job Message Templates, EOJ type code. The sequence requirements of a control file are a) all records of a type (excluding comment records) must be together, b) the order (excluding comment records) must be as defined above, and c) comment records must not appear within any group.

Records selected for processing must match on both type code and number. That is, each processing routine supplies a key code and a key number, both three character fields. These fields must match a control record's type code and number, respectively, for the record to be processed. For purposes of determining a match on number, each byte of a control record's number field must be either blank or match the corresponding byte in the key number field. For HSK, CTL, and EOJ control records, the key number supplied is either 001 or 002, depending upon the maner in which the MCCRESSA program root was loaded, i.e., $L or $JOBUTIL, respectively. For ERR control records, the key number is the terminal error message number.

2.3. Message Templates

The body of each message template selected for processing generates a line of text upon the invoking terminal. This line of text may be up to 80 bytes in length. It may include both text from the body of the message template and text from program variables. The inclusion of text from program variables is accomplished by a parameter substitution algorithm and a stack of words supplied by the calling program for the paticular type of message template being processed.

Basically, each byte of the body of a message template is moved to the output message area until the end of either the body of the message template or the output area is reached or a parameter begin character (backslash '\') is found. If a parameter begin character is found, the body of the message template is then scanned until a parameter end character (period '.') is found. The parameter end character is required. The intervening characters constitute the parameter which must be of the form a) Cn, b) i, c) iLn, d) +i, or +iLn. The parameter is then processed and the above process is repeated.

The interpretation of the characters used above in the definition of the different forms a parameter might take is as follows.

C	Reset output column location
n	Simple numeric of from 1 to 80
i	Index of a stack word
L	Length of the parameter text specified
+	Parameter is character string

In case a) the parameter resets the output column location to n if it is not already greater than n. In all other cases the ith stack word is referenced. To be logically correct, the stack must have at least i entries. No check can be made that this is true. In case b) the referenced stack word is a simple numeric which is converted to character form and then moved to the output area without leading zeros. In case c) the referenced stack word is a simple numeric which is converted to character form and then right justified without leading zeros in the next n output locations. In case d) the referenced stack word is the address of a TEXT area whose length is given in the byte immediately preceding the TEXT area. The TEXT characters so located are moved to the output area. In case e) the referenced stack word is the address of a TEXT area whose length is, for this use, n. The TEXT characters so located are moved to the output area. In cases b) through e) the process is terminated if the end of the output area is reached.

If the parameter found does not conform to one of the above forms, it is considered not a parameter and its defining characters are included as text in the output display. This can occur because i or n are not strictly numeric, other nonnumeric characters than C, L, or + are present, or the parameter is not terminated with a period.

2.4. Program Control Value Records

Program control value records, CTL type code, each contain from 1 to 65 characters of program control data. There must be exactly one selected program control value record for each word pair in the CPCTLSTK, no more, no less. For each such record, the number of characters given in word one of the stack word pair are moved to the program area located by the second word. Any characters in the body of the control record to the right of those moved are ignored and may be used for clearifying comments, e.g., type of control data.

The match of control value records to CPCTLSTK word pairs is positional. If all CTL control records have a blank number field, the control file string of CTL control records is also the processed string of control records. If, however, some of these records have non blank number fields, they must appear in pairs with an adjacent 002 for each 001, the only allowed values. That is because the selected set is a subset of the control file set and the strict match between the selected set and CPCTLSTK word pairs must be maintained. It is through this selection process that a program may be set up to operate with different defaults depending upon the manner in which it is loaded.

2.5. PGMCTL Error Messages

PGMCTL displays its own originated error messages only when the contents of the control file are found to be in error or there is an input/output problem associated with the control file. These two classes are presented separately below.

2.5.1. Control File Content Errors

The control file content errors are a direct result of missing or extra control file records. That is, both a HSK and EOJ linkage must generate at least one message. Also, the number of word pairs in the CPCTLSTK must exactly equal the number of selected CTL recrods. When one of these conditions arises, PGMCTL issues a return code of 16 to the calling MCCRESSA functional program root after issuing one of the messages listed below, as appropriate.

NO END-OF-JOB MESSAGES GIVEN - ERROR EOJ.

NO HOUSEKEEPING MESSAGES GIVEN - ERROR EOJ.

NOT ALL CONTROL AREA RECORDS USED - ERREOJ.

NOT ALL CONTROL AREAS FILLED - ERREOJ.

2.5.2. Control File Input/Output Errors

When an input/output error occurs PGMCTL issues the following message:

ERROR PROCESSING MEMBER mmmmmmmm.vvvvvv - cccc

where mmmmmmmm is the control file member name, vvvvvv is the EDX volume name from which the calling MCCRESSA functional program was loaded, and cccc is one of the messages below. It then issues a return code of 16 to the calling MCCRESSA functional program root.

DSN BEGINNING WITH $$ ILLEGAL

I/O ERROR DURING OPEN

MEMBER NOT FOUND

NOT DATA TYPE

READMSG1
(simple I/O error reading control file)

VOLUME NOT FOUND OR UNUSEABLE

VOLUME NOT INITIALIZED

3. Set Program Parameters (SETPRM)

SETPRM is a command interpreter program designed specifically for the entry of program control information. Typically this program is loaded one or more times by a MCCRESSA program root. It will transfer free form control information from a control file to fixed form control information in memory. The purpose of supplying program control information in this manner is that it allows the user interface to be determined by a relatively simple control file rather than embedded in one or more overlays of a MCCRESSA program. In fact, a single version of a MCCRESSA program, in conjuction with one of several control files, may a) operate with each program control area filled interactively to meet an ad hoc requirement, b) operate with a select set of program control areas filled interactively to meet a normal operational requirement, or c) operate with no user interaction to meet a common requirement in a predetermined manner.

The entry of program control information into the various control areas in memory is directed by commands within the control file which reference keywords defined by the MCCRESSA program root. Commands within the control file may move control information from one area to another, request interactive entry of control information, alter the sequence the control file is processed, and interrupt control file processing to return to the program root and, when next called, resume control file processing at the point of interruption. Furthermore, all commands may be conditionally executed based upon the current contents of a control area.

3.1. Program Parameters

Two parameters, supplied by the loading program, are required by SETPRM. Since SETPRM is a COBOL program, the address of a stack of words must be passed where each word in the stack is the address of a COBOL referenced parameter. The two required parameters are the control area and the keyword table.

3.1.2. Control Area

The control area, which must begin on a word boundary, is referenced by SETPRM primarily as an array of characters (PIC (X) OCCURS) with undetermined dimension. The first 36 bytes are, however, also referenced as a COBOL structure and therefore have a prescribed meaning. Figure 3.1 gives this COBOL structure code copied into SETPRM and any other COBOL program referencing the standard SETPRM control area. This code is usually stored on MCCRESSA program development diskettes under the name of STDPARM. In application programs, this prescribed structure may be extended with application dependent control areas by immediately following the COPY STDPRM statement with area definitions at levels 2 and higher. The only restriction is that all additional areas referenced via SETPRM must be defined with the picture definition characters X or 9 (9 without S or COMP SYNC).

```
      *  STANDARD CONTROL PARAMETER BEGINNING - VERSION 1.1 - 05/31/84
       1   CONTROL-PARM.
 0      2   RETURN-CODE                 PIC S9999 COMP SYNC.
 0      2   DIAG-CNT                    PIC S9999 COMP SYNC.
10      2   PARM-SELECT-MAX             PIC S9999 COMP SYNC.
        2   PARM-DATE                   PIC X(8).
        2   PARM-TIME                   PIC X(8).
        2   DIAG-RESP-SW                PIC X.
        2   PARM-ID-SW                  PIC X.
        2   CF-EOF-SW                   PIC X.
        2   PARM-SELECT-TEST            PIC X.
        2   PARM-SELECT-CHAR            PIC X OCCURS 10 TIMES.
```

STDPARM Copy Code

Figure 3.1

MCCRESSA program roots, written in EDL, which load SETPRM each require an EDL definition of the beginning of the standard SETPRM control area. Such a definition is stored in EDLLIB under the name ROOTPARM (Figure 3.2). All application dependent control areas referenced by SETPRM are located by their location relative to the beginning of the control area, i.e., through a COBOL character array whose first element is at SPPARM. This scheme makes 32767 bytes accessible. Thus, it is standard practice to include ROOTPARM near the beginning of a MCCRESSA program root making all subsequent areas of the program accessible (no root is greater than 32767 bytes).

```
********************* ROOTPARM -> STANDARD PROGRAM PARAMETER - 06/06/84
          ALIGN
SPBASE    EQU       *-1           BASE FOR ADDRESS TO INDEX CONVERSION
SPPARM    EQU       *             ALL OF PROGRAM AVAILABLE TO PARAMETER
SPRTNCOD  DATA      F'0'          COMMON PROGRAM RETURN CODE AREA
SPDIAGCT  DATA      F'0'          COMMON PROGRAM DIAGNOSTIC COUNT
SPSELCNT  DATA      F'10'         SIZE OF SELECT CODE AREA (SPSELCOD)
SPRUNDAT  DATA      CL8' '        RUN DATE AREA MM/DD/YY
SPRUNTIM  DATA      CL8' '        RUN TIME AREA HH:MM:SS
SPDURRSW  DATA      CL1'Y'        DIAGNOSTIC USER RESPONSE REQUIRED SWITCH
SPDPIDSW  DATA      CL1'N'        DISPLAY PROGRAM ID SWITCH
SPCTLEOF  DATA      CL1'N'        'Y' -> CONTROL FILE AT EOF
SPRESERV  DATA      CL1' '        RESERVED FOR SELECT CODE SCAN ROUTINE
SPSELCOD  DATA      CL10' '       CONTROL CARD SELECT CODE AREA
*************** INSERT COBOL REFERENCED PARM EXTENSION DEFINITIONS HERE
```

ROOTPARM.EDLLIB

Figure 3.2

Some application programs loaded by MCCRESSA program roots are themselves secondary roots, also written in EDL. If a secondary root references the standard SETPRM control area, it requires a similar control area definition. Such a definition is stored in EDLLIB under the name ROOTEQU (Figure 3.3). Any application dependent appendages to the standard SETPRM control area referenced by a secondary root must be supplied in a like manner.

```
************** ROOTEQU -> STANDARD PROGRAM PARAMETER EQUATES - 01/28/85
SPPARM   EQU       0
SPRTNCOD EQU       SPPARM       COMMON PROGRAM RETURN CODE AREA
SPDIAGCT EQU       SPRTNCOD+2   COMMON PROGRAM DIAGNOSTIC COUNT
SPSELCNT EQU       SPDIAGCT+2   SIZE OF SELECT CODE AREA (SPSELCOD)
SPRUNDAT EQU       SPSELCNT+2   RUN DATE AREA MM/DD/YY
SPRUNTIM EQU       SPRUNDAT+8   RUN TIME AREA HH:MM:SS
SPDURRSW EQU       SPRUNTIM+8   DIAGNOSTIC USER RESPONSE REQUIRED SW
SPDPIDSW EQU       SPDURRSW+1   DISPLAY PROGRAM ID SWITCH
SPCTLEOF EQU       SPDPIDSW+1   'Y' -> CONTROL FILE AT EOF
SPRESERV EQU       SPCTLEOF+1   RESERVED FOR SELECT CODE SCAN ROUTINE
SPSELCOD EQU       SPRESERV+1   CONTROL CARD SELECT CODE AREA
******************** INSERT COBOL REFERENCED PARM EXTENSION EQUATES HERE
```

ROOTEQU.EDLLIB

Figure 3.3

The standard SETPRM control area beginning contains a return code number (one word), a diagnostic counter (one word), a select field size number (one word), a date area (8 bytes - MM/DD/YY), a time area (8 bytes - HH:MM:SS), a diagnostic user response required switch (1 byte), a program id display switch (1 byte), the control file EOF switch (1 byte), a reserved byte, and the select field (length as noted in word three - presently 10 bytes).

3.1.3. Keyword Table

The keyword table, which must begin on a word boundary, contains entries of the form n1 (two byte binary number), n2 (one byte binary number), n3 (one byte binary number), and keyword (eight byte character string). The first entry, a control entry, uses n1 as the table size, n2 and n3 taken together as the control file current records read count (initially zero), and keyword as the table scan work area which will force a found condition at I=0 when scanning the table from top (I=n1) to bottom (I=1) for a keyword not in the table. The second and remaining entries, referenced by index values of 1 and higher, each define an area in the control parameter which may be referenced by keyword. Keywords may be from one to eight characters in length. Those keywords beginning with a pound sign (#) are considered to define a numeric area. All other keywords are considered to define an alphameric (character) area. The referenced area is located by n1 and is of length n3. The currently used length for character areas and the number of

implied decimal places for numeric areas is given by n2.

The only COBOL program that references the keyword table is SETPRM. Figure 3.4 gives the COBOL definition of this table. As with most parameter definitions, the diminsion of an array is meaningless. SETPRM gets the size of KEYWORD-TABLE from LAST-KEYWORD-I which is filled during root housekeeping.

```
1   KEYWORD-PARM.
 2  LAST-KEYWORD-I          PIC S9999 COMP SYNC.
 2  CF-RCD-CNT              PIC S9999 COMP SYNC.
 2  TEST-KEYWORD            PIC X(8).
 2  KEYWORD-TABLE           OCCURS 2 TIMES.
  3  PARM-LOC               PIC S9999 COMP SYNC.
  3  PARM-LNG               PIC S9999 COMP SYNC.
  3  PARM-LNG-CHARS REDEFINES PARM-LNG.
   4  PARM-LNG-CHAR1        PIC X.
   4  PARM-LNG-CHAR2        PIC X.
  3  PARM-KEYWORD.
   4  PARM-KEYWORD-1        PIC X.
   4  FILLER                PIC X(7).
```

Keyword Table Definition

Figure 3.4

MCCRESSA program roots that load SETPRM must contain the definition of the keyword table. Two copy code data sets, resident in EDLLIB, define the beginning and end of the required keyword table. These are ROOTKWRD (Figure 3.5) and ROOTEND (Figure 3.6), respectively. To use these copy code data sets, it is required that the keyword table be defined at the end of the program. That is, the COPY ROOTKWRD statement is coded, the definition of any application dependent keywords are coded, and then the COPY ROOTEND statement is coded as the last statement in the program.

For proper operation, the number of keyword entries must be given in KEYWORDN. In order to make the setting of this value automatic, each MCCRESSA program root defining the keyword table must, in its housekeeping routines, include code similar to that given in Figure 3.7. This code will automatically enter the required value, i.e., the MCCRESSA programmer need not remember to update the number of keywords in the table when modifying the program.

```
****************** ROOTKWRD - SETPRM PARAMETER/KEYWORD TABLE - 06/06/84
SETPRMPA DATA      A(SETPRMP)
         DATA      F'0'
SETPRMP  DATA      A(SPPARM)
         DATA      A(KEYWORDP)
********************************************** STANDARD KEYWORD TABLE
         ALIGN
KEYWORDP EQU       *
KEYWORDN DATA      F'006',F'000',C'        ' KEYWORDCNT,CARDCNT,RESERVE
KEYWORD1 DATA      A(SPRUNDAT-SPBASE),F'0008',C'DATE    '  KEYWORD  1
         DATA      A(SPRUNTIM-SPBASE),F'0008',C'TIME    '           2
         DATA      A(SPDURRSW-SPBASE),F'0001',C'WAITRESP'           3
         DATA      A(SPDPIDSW-SPBASE),F'0001',C'DISPID  '           4
         DATA      A(SPCTLEOF-SPBASE),F'0001',C'CTLEOF  '           5
         DATA      A(SPSELCOD-SPBASE),F'0010',C'CTLSEL  '           6
* INSERT ADDITIONAL KEYWORD ENTRIES HERE
```

ROOTKWRD.EDLLIB

Figure 3.5

```
***************************** ROOTEND -> END OF KEYWORD TABLE - 06/06/84
KEYWORDL EQU       *-KEYWORD1          LENGTH OF KEYWORD TABLE
KEYWORDC EQU       KEYWORDL/12         NUMBER OF KEYWORD ENTRIES
*****************************************************************************
         ENDPROG
         END
```

ROOTEND.EDLLIB

Figure 3.6

```
START    MOVE      KEYWORDN,+KEYWORDC       NUMBER OF KEYWORD ENTRIES
```

Code to Set Number of Keyword Entries

Figure 3.7

3.2. Control File

The control file is assumed to have been created by a typical EDX editor. As such, 128 byte logical records are packed two to the 256 byte EDX disk record. Furthermore, only the first 80 bytes of a logical record are considered to be data. This 80 byte data field is referred to here as a control card which is logically divided into five fields as described below.

3.2.1. Control Card

Control card column one is reserved as a carriage control field. This field may be used to format a card image printout of a control file with a simple utility routine. Control cards having a '1', 'T', or 'C' in column one are ignored by the SETPRM program.

Control card column two is a select code field. Any character, including blank, is valid in this position. If this field is blank the card is included in the processing unconditionally. If this field is non-blank the card is included only if the select code character also appears as one of the control area's select characters, i.e., matches a PARM-SELECT-CHAR. The contents of the control area select field may be set by both the calling program and the control card stream.

Control card column three is a control card type code. Valid type codes are blank, '+', and '*'. Control cards with a type code of '*' are considered to be comments and receive no processing. For the distinction between blank and '+' see Section 3.2.2, below.

Control card columns 4-72 are the control card's data area. For further discussion of this field's use see Section 3.2.2, below.

Control card columns 73-80 are the editor's sequence area. The only use SETPRM makes of this field is in connection with diagnostic messages. If a control card error is found, this field is included in the ensuing diagnostic message to facilitate user correction.

3.2.2. Control Record

At this point in the discussion the concept of a control record, as differs from a control card, needs to be introduced. A control record may consist of data fields from one to ten control cards, the first of which has a type code of blank. All remaining data fields to be considered as part of the same control record must each be from successive control cards having a type code of '+'. The data field (69 characters) of each included control card may be considered juxtaposed to its neighbors giving a contiguous control record area of up to 690 characters. Each control record processed contains exactly one unconditional command.

3.2.3. Logical Words

Control records contain sets of logical words. A logical word may be defined as any string of up to 255 characters delimited by one or more blanks or a special character. The present special characters which may delimit a logical word are '+', '=', '(' or either the single or double quote characters. The quote characters define literals which may contain any character other than the defining quote character. Thus, a character string which begins with a single quote is terminated by the next occurence of the single quote or the end of the current control record, which ever occurs first. Similarly with the double quote. For example, the following character string is considered to be a logical word of 31 characters.

"A logical word of 31 characters"

Logical words beginning with a left parenthesis must be terminated by a right parenthesis and are of the form KEYWORD,N,M with the commas optional depending on the inclusion of the positional parameters N and/or M. The simplest form, (KEYWORD), implies the current value of KEYWORD rather than the character string '(KEYWORD)'. For character keywords, N,M implies the retrieval of a substring of the current value, i.e., N is the index of the first character retrieved and M is the index of the last character retrieved. The default N and M values are one and the current length, respectively. For numeric operands, N is the maximum number of decimal places and M is the maximum width of the retrieved value, exclusive of the decimal point if required. The actual value retrieved has no leading zeros to the left of the units position and no trailing zeros after the decimal point. The decimal point is only included if there is a non-zero fraction to the retrieved value. All digits not addressed by a supplied N and or M value are ignored (truncated).

Further, two or more adjacent logical words retrieved as described above, may be joined together (juxtaposed) before processing to form a single logical word. This is done by separating them with a '+' (the concatenation symbol), intervening blanks ignored. For example, the following two lines represent equivalent SETPRM logical words for processing.

```
'34 characters in this logical word'
34 + ' characters ' + in + ' this logical word'
```

Similarly, the following line represents a logical word containing each quote symbol once.

```
'"'+"'"
```

3.3. Commands

Each control record begins with a one to eight character word which is interpreted as a command to this program. A command determines how the remainder of a control record will be processed. The recognized commands are discussed below in alphabetical order.

3.3.1. DEFAULT

Each command begins with an edit control area being initialized. This area is then modified as a control record is processed. The DEFAULT command processes edit control keywords and then returns the edit control area to the initializing area. Thus the edit control defaults may be changed. For a list of edit control keywords and their defaults see Section 3.4.

3.3.2. DISPLAY

The DISPLAY command displays each logical word from the control record. Due to a restriction of COBOL the length of each displayed line is 60 characters. Logical words shorter than 60 characters are padded with blanks and those longer than 60 characters are truncated.

3.3.3 END

The END command terminates execution of the SETPRM program and returns control to the calling program. This action is similar to an end-of-file condition on the Control File. A reason for using an END card might be to allow the calling program to continue, possibly calling a program to be controlled by the control information just processed. The calling program may later call the SETPRM program for further control card processing. The number of the last control card read is maintained in the calling program's area and is thus available for any following call. When SETPRM is next entered the control file is repositioned to the point at which the program was interrupted.

3.3.4. IF

The IF command requires an alphameric relational expression to follow which, in turn, is followed by another SETPRM command. If and only if the relational expression is true is the following SETPRM command executed. See Section 3.6 for an explanation of relational expressions.

3,3.5. IFN

The IFN (if not) command requires an alphameric relational expression to follow which, in turn, is followed by another SETPRM command. If and only if the relational expression is not true is the following SETPRM command executed. See Section 3.6 for an explanation of relational expressions.

3.3.6. JUMP

The JUMP command may have one or two operands. The first (required) operand is interpreted to be a label. The second (optional) operand must be a numeric keyword. If the second operand is supplied, the current card number is stored in that area. In any case, the JUMP command scans each following control record for a LABEL record having an operand equal to the first JUMP operand. If no such LABEL record is found,the control file is closed, reopened, and scanned again from the first record. If a not found condition still exists a diagnostic message is displayed and the program return code is set to four. Otherwise, processing continues at the control record immediately following the matched LABEL record.

3.3.7. LABEL

The LABEL command has one operand which may be the object of a JUMP command as described above. When encountered in normal processing, the LABEL command is treated as a no-operation record.

3.3.8. MARK

The MARK command inserts flags into an alphameric keyword's value. The format of the MARK command is

MARK keyword=flag n1 n2 n3 ... nn

where keyword and flag are logical words, keyword does not begin with a pound sign (not a numeric keyword), and all remaining operands are numeric. The numeric operands must each be greater than zero and less than the maximum length of the operand identified by keyword. If flag is not supplied, the letter X will be assumed. If flag is supplied, the first character will be the flag character. In operation, a flag is placed in each position ni. If the current length of the marked operand is less than ni, it is set to ni.

3.3.9. NIF

The NIF (numeric if) command requires a numeric relational expression to follow which, in turn, is followed by another SETPRM command. If and only if the relational expression is true is the following SETPRM command executed. See Section 3.6 for an explanation of relational expressions.

3.3.10. NIFN

The NIFN (numeric if not) command requires a numeric relational expression to follow which, in turn, is followed by another SETPRM command. If and only if the relational expression is not true is the following SETPRM command executed. See Section 3.6 for an explanation of relational expressions.

3.3.11. NO

The NO command requires one operand and a following SETPRM command. The single operand is the prompt for a user response for which the only acceptable answers are Yes or No. If the user responds NO (or N) the following SETPRM command is executed. Otherwise, the remainder of this control record is ignored.

3.3.12. RESET

The RESET command may have any number of operands, each consisting of from one to eight characters in length. Each operand is interpreted to be a keyword whose value is to be reset. Character keywords are reset to spaces and numeric keywords are reset to zeros.

3.3.13. RETURN

The RETURN command requires a single operand. This operand must be a numeric keyword (begin with a pound sign). The contents of the keyword's area is retrieved and the control file positioned such that the control record identified by its positional location in the file will be the next record processed. This command may be used in conjuction with the JUMP command with two operands to effect a jump and link operation. Also, it may be used in conjuction with the SAVE command to cause the same effect as a JUMP and LABEL pair of commands.

3.3.14. SAVE

The SAVE command requires a single operand. This operand must be a numeric keyword (begin with a pound sign). The current control card number is stored in the keyword's area. The current control card number is actually the number of the next control card to be processed.

3.3.15 SET

The SET command assigns a value to a keyword. The basic form of this command is

SET keyword=operand

where keyword and operand are both logical words. Keyword must consist of from one to eight characters and be one of the keywords known to the calling program. The assignment is controlled by:

1) The type of keyword, i.e., if the keyword begins with a # the assignment is numeric, otherwise the asignment is alphameric.

2) The edit control area. This area is initialized before command processing. The SET command may alter this area (such alterations apply to the current command's execution only).

3) The keyword's description, i.e., its maximum length and, for numeric operands, the number of decimal places, and for alphameric operands, the current length.

Following the operand may be any number of edit and justification control operands as listed in Sections 3.4 and 3.5, respectively. These edit and justification control operands are processed (the edit/justification control area is updated) before any assignment is made.

3.3.16. USER

The USER command operates similar to the SET command with the exception that the value assigned to the keyword is taken from the terminal keyboard rather than the control record. The command's basic format is

USER keyword=control PROMPT=message

where keyword is the keyword to receive the assigned value, control is a code which controls the required user response, and message is the text displayed before the keyboard is read. The valid values for control and their meanings are as follows:

C -> Convert all lower case alphabetic characters to upper case. The default is to use each character as entered.

F -> Force a new user response. The default is to use a previously supplied response if one exists.

L -> Line response, accept the full user response. The default is word, i.e., return only the first group of characters delimited by one or more spaces.

One or more of these characters may be supplied in any order. If the default conditions are desired, no control operand is required and keyword need not be followed by an equal sign.

If PROMPT=message is missing then the prompt is simply 'keyword?'. If a stored user response is used, no prompt is displayed on the screen.

3.3.17. YES

The YES command requires one operand and a following SETPRM command. The single operand is the prompt for a user response for which the only acceptable answers are Yes or No. If the user responds YES (or Y) the following SETPRM command is executed. Otherwise, the remainder of this control record is ignored.

3.4. Edit Control

Edit control keywords are used to set the various editing options before any assignment of by a SET or USER command. In the absence of an explicit edit control keyword on one of these statements the default value applies. Default values in effect when SETPRM is loaded are identified below. These defaults may be changed by use of the DEFAULT command as explained above.

3.4.1. CLEAR/NOCLEAR

These edit control keywords have meaning for character keywords only. CLEAR causes the current length of the object keyword to be set to zero and its contents to be set to spaces before any assignment is processed. NOCLEAR inhibits the clear action. The default is CLEAR.

3.4.2. FLAG/FLAGNOT

These edit control keywords cause the replacement of each character in the assignment argument by either a space or the letter X, thus the object of any subsequent assignment operation must be a character

keyword. The format is as follows:

FLAG=string

The flag operand string may consist of from one to fifteen non-blank characters. In operation there are two switches, FLAG and FLAGNOT. The switch that matches the edit control keyword is set to the character X and the other switch is set to a space. The assignment argument is scanned, character by character. Each character is replaced by FLAG if it is contained in the flag operand and by FLAGNOT if it is not contained in the flag operand. The subsequent assignment is processed as any other MOVE. The default is a simple MOVE statement.

3.4.3. JUST

This edit control keyword is of the form

JUST=operand

where operand is any of the five discussed in Section 3.5, below.

3.4.4. MOVE

This edit control keyword is the default (alternatives are FLAG and FLAGNOT). For numeric keywords the assignment process checks for a strict numeric argument, aligns the given decimal point with the implied decimal point, and assigns the argument to the object keyword. For character keywords the assignment process begins with a conditional clearing of the object keyword (see CLEAR), the assignment of the argument to the object keyword with any requested justification, and the setting of the resulting current length.

3.5. Justification Control

The justification control keywords, discussed below, are used to control the alphameric assignment of values to operands for the SET and USER commands. Each of the justification keywords are mutually exclusive. The last supplied value holds for the current assignment. If no justification keywords are given, the default justification applies. The default justification may be set by the DEFAULT command. Initially the default is Left. The initial character of the words given below is the valid operand.

3.5.1. Center

The current argument is centered based upon the maximum size of keyword without regard to the current contents. The assignment process is an overlay process and any positions outside the assignment area are not disturbed. If the position of the last character assigned is beyond the current length, the current length is reset.

3.5.2. Left

The current argument is assigned beginning in the first position of the keyword's area. If the assigned argument is longer than the current length, the length of the current argument is set as the current length.

3.5.3. Next

The current argument is assigned beginning in the first position to the right of the current length of the keyword's area. The new current length is set to reflect the additional characters. If the current length is initially zero, this justification defaults to Left.

3.5.4. Position

The current argument is assigned to the keyword's area beginning in the position supplied by the numeric immediately following the justification code P. If the last assigned character position is to the right of the current length, that position is set as the current length.

3.5.5. Right

The current argument is right justified in the keyword's area. The maximum length is set as the current length.

3.6. Relational Expressions

Relational expressions are used for the IF, IFN, NIF, and NIFN commands to give conditional execution of a command following the relational expression. The general format of a relational expression is

```
A R B [[O R B]...]
```

where A and B are logical words to be compared, R is a relation, and O is a boolean operator. The brackets imply optional extensions and do not appear in an expression.

The recognized relations are

```
= < > <= >= <>
```

which are read equal, less than, greater than, less than or equal, greater than or equal, and unequal, respectively. The order of characters is ignored for those relations identified by two characters.

The boolean operators are | for 'or' and & for 'and'. Compound expressions involving both types of boolean operators are nested such that adjacent 'and' operations are surrounded by implied parentheses. No parentheses may appear in the expression.

In the notation above, there is one A operand and one or more B operands. Logically, each compare is between the single operand A and each operand B. The compare is alphameric for IF and IFN commands and

numeric for NIF and NIFN commands. Either the A or B operand in each compare must involve a keyword in some manner or else the results are predetermined.

3.7. Program Messages

SETPRM issues informative, warning, and terminal diagnostic messages.

3.7.1. Informative Diagnostic Messages

Informative diagnostic messages are provided for debugging purposes and, as such, are conditional upon the value of PARM-ID-SW (byte 24) of the standard control area. If this switch has a value of 'Y', SETPRM issues informative messages during both housekeeping and end-of-job processing. These two messages are listed below.

```
SET CONTROL PARAMETER - VERSION 2.0
END OF SET PARAMETER PROGRAM.
```

3.7.2. Warning Diagnostic Messages

Warning diagnostic messages are issued for those errors noted during the processing of the control file that do not preclude further execution of SETPRM. Since continuing most probably will not produce the desired results, the RETURN-CODE (word 1) of the control parameter is set to four so that the MCCRESSA functional program may terminate execution after SETPARM when one or more warning diagnostic messages have been issued. In this manner, more than one control file error may be located at a time.

To facilitate control file debugging, warning diagnostic messages are issued by a common routine that also displays the current position of the control file and, conditionally, waits for a user response before proceeding. This standard diagnostic display is depicted in Figure 3.8.

```
1     ***** PROGRAM DIAGNOSTIC *****

2     mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm

3     SCAN AT CARD ssssssss COLUMN pp AS FLAGGED BELOW.

4     ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
5                                       :

6     PRESS ENTER TO CONTINUE.
```

Warning Diagnostic Message Format

Figure 3.8

When interpreting Figure 3.8 the following should be noted:

a) The initial number at the beginning of each non-blank line is not part of the message as displayed by SETPRM but rather is included here in order to aid in the discussion.

b) Lower case letters indicate variable information fields filled in when displayed.

c) The m's in line two represent the text of the diagnostic message displayed.

d) Line three gives the editor sequence number (ssssssss) and scan position (pp) of the control card currently being processed. The warning diagnostic message issued is generally the result of the logical word immediately preceeding this location in the control file.

e) Line four gives up to 60 characters of the current control card. The card columns displayed surround the position noted in line three above.

f) Line five contains a single colon to flag the position noted in line three above.

g) Line 6, which requires a user response, is displayed only if the value of DIAG-RESP-SW (byte 23) of the standard control area is a 'Y'.

The possible warning diagnostic messages, along with their COBOL tags, are listed in Figure 3.9. Each of these messages are discussed in detail below.

3.7.2.1. KEYWORD BEGINNING 'XXXXXXXX' TOO LONG.

A keyword was expected and the next logical word found exceeded eight characters in length. The keyword expected may be a SETPRM command (see Section 3.3), an edit control keyword (see Section 3.4), or a parameter keyword (see Section 3.1.3).

3.7.2.2. INVALID NUMERIC OPERAND.

A logical word which is not numeric is being assigned to a numeric program control area. A numeric logical word may contain only numerics, a decimal point, and commas (ignored).

3.7.2.3. PARM KEYWORD 'XXXXXXXX' NOT FOUND.

A logical word was interpreted to be a parameter keyword and was not found in the keyword table. This keyword was to have been used as the object of a MOVE, FLAG, or FLAGNOT command or as a source for constructing another logical word.

Tag	Text
M02	KEYWORD BEGINNING 'XXXXXXXX' TOO LONG.
M03	INVALID NUMERIC OPERAND.
M04	PARM KEYWORD 'XXXXXXXX' NOT FOUND.
M05	CONTROL KEYWORD 'XXXXXXXX' UNKNOWN.
M06	FUNCTION KEYWORD 'XXXXXXXX' UNKNOWN.
M07	FLAG OPERAND 'XXXXXXXXXXXXXXX' TOO LONG - TRUNCATED.
M08	FLAG OPERAND MISSING.
M09	JUST OPERAND 'XXXXX' INVALID - L ASSUMED.
M11	OPERAND TOO LONG.
M12	TOO MANY COMMAS IN KEYWORD SUBSTRING EXPRESSION.
M13	'X' INVALID AFTER COMMA IN KEYWORD SUBSTRING EXPRESSION.
M14	ZZZ9 INVALID AFTER COMMA IN KEYWORD SUBSTRING EXPRESSION.
M15	RESPONSE CONTROL CHARACTER 'X' UNKNOWN.
M17	NO PROMPT FOR YES/NO COMMAND, IGNORED.
M18	CANNOT MARK NUMERIC KEYWORD.
M19	INVALID MARK POSITION.
M20	INVALID RELATION.
M21	PROGRAM ERROR - EC-RTN-I = 9999.
M22	INVALID CARD NUMBER SAVE KEYWORD.
M23	INVALID RETURN CARD NUMBER.
CF-M05	TYPE CODE (COL 3) INVALID.

Warning Diagnostic Messages

Figure 3.9

3.7.2.4. CONTROL KEYWORD 'XXXXXXXX' UNKNOWN.

A logical word was interpreted to be an edit control keyword (see section 3.4) and was not found in the edit control keyword table.

3.7.2.5. FUNCTION KEYWORD 'XXXXXXXX' UNKNOWN.

A logical word was interpreted to be a SETPRM command and was not found in the table of command keywords.

3.7.2.6. FLAG OPERAND 'XXXXXXXXXXXXXXX' TOO LONG - TRUNCATED.

A logical word was interpreted to be the edit control operand of either FLAG or FLAGNOT and exceeds the maximum length of fifteen characters (see Section 3.4.2).

3.7.2.7. FLAG OPERAND MISSING.

The edit control keywords FLAG and FLAGNOT require an operand which was not found (see Section 3.4.2).

3.7.2.8. JUST OPERAND 'XXXXX' INVALID - L ASSUMED.

A logical word is the object of the edit control keyword JUST and is not recognized as a valid operand (see Section 3.5).

3.7.2.9. OPERAND TOO LONG.

A logical word is being assembled and its length exceeds 255 characters. To be a warning diagnostic message the assembly at this point is from a parameter keyword value. If the assembly is from the control record area this message is terminal.

3.7.2.10 TOO MANY COMMAS IN KEYWORD SUBSTRING EXPRESSION.

A logical word in the control record which begins with a left parenthesis is followed by text which contains more than two commas. This is not a valid form for a parameter keyword substring expression (see Section 3.2.3).

3.7.2.11. 'X' INVALID AFTER COMMA IN KEYWORD SUBSTRING EXPRESSION.

A non-numeric has been located in the N or M area of a parameter keyword substring expression (see Section 3.2.3).

3.7.2.12. ZZZ9 INVALID AFTER COMMA IN KEYWORD SUBSTRING EXPRESSION.

The numeric N or M of a parameter keyword substring expression is greater than the maximum length of a parameter keyword (see Section 3.2.3).

3.7.2.13. RESPONSE CONTROL CHARACTER 'X' UNKNOWN.

A USER command is being processed. The logical word interpreted as the response control word contains characters other than 'C', 'F', or 'L' (see Section 3.3.16).

3.7.2.14. NO PROMPT FOR YES/NO COMMAND, IGNORED.

A YES or NO command must be followed by a logical word which is intrepreted as a prompt. Nothing follows the command so no user response is requested and this command defaults to a no operation.

3.7.2.15. CANNOT MARK NUMERIC KEYWORD.

The object of a MARK command is a numeric parameter keyword (begins with a pound sign). An alphameric parameter keyword is required (see Section 3.3.8).

3.7.2.16. INVALID MARK POSITION.

A position operand for a MARK command has been found that is either not numeric or has a value greater than the length of the parameter keyword referenced (see Section 3.3.8). This operand and any remaining operands are ignored.

3.7.2.17. INVALID RELATION.

A relational expression is being scanned. A recognized relation was not found when required (see Section 3.6).

3.7.2.18. PROGRAM ERROR - EC-RTN-I = 9999.

The edit routine index is set to a value greater than provided in the program. This can only occur if the another edit function is improperly added to the program. The MCCRESSA programmer should be notified. The contents of the control file cannot cause this error.

3.7.2.19. INVALID CARD NUMBER SAVE KEYWORD.

A parameter keyword is either missing, not numeric, or is not found in the parameter keyword table. It is the object of a SAVE command (see Section 3.3.14).

3.7.2.20. INVALID RETURN CARD NUMBER.

The logical word following a RETURN command contains a value exceeding 9999 (see Section 3.3.13). No operation is performed.

3.7.2.21. TYPE CODE (COL 3) INVALID.

A card following the one displayed, or one of its continuation cards, has an invalid type code (see Section 3.2.1). This card is ignored.

3.7.3. Terminal Diagnostic Messages

Terminal diagnostic messages are issued by setting the return code (word one) of the standard control area and returning to the calling program root. The calling program root should then invoke PGMCTL to display the selected message(s). Listed below are the card images which must appear in the program's PGMCTL control file for the two possible SETPRM terminal diagnostic messages.

3.7.3.1. ERR004 WARNING DIAGNOSTICS NOTED.

3.7.3.2. ERR012 SEVERE ERROR DIAGNOSTICS NOTED.

4. MCCRES Master File Information (MFINFO)

MFINFO reads a MCCRES master file header record, either evaluation or nomenclature, and maps it into a 339 character parameter area in memory. Typically, this program is loaded by a MCCRESSA program root before loading the primary functional program. If the MCCRESSA program processes more than one MCCRES master file, MFINFO would be loaded once for each master file processed. Multiple master file header parameter areas may be utilized for concurrent processing of multiple master files.

MFINFO requires five parameters as follows:

a) Control Parameter;
b) Master File Header Parameter, Part 1;
c) Master File Header Parameter, Part 2;
d) Program Error Parameter; and
e) Program Identification Parameter.

The Control Parameter is the standard SETPRM control parameter which is normally copied into a MCCRESSA program using STDPARM copy code (see Figure 3.1). The Master File Header Parameter will be treated here as one parameter because it is that for most programs. In MFINFO it is divided into two parts due to the COBOL requirement that only level one areas may be referenced by USING statements. Finally, the Program Error and Program Identification parameters are required inter-program communication parameters for MCCRESSA programs utilizing PGMCTL (see Section 2.).

4.1. Master File Header Parameter

Two definitions of the MCCRES master file header paramater area are required to utilize MFINFO. These are the EDL definition which must be included in AN EDL root and the COBOL definition which must be included in a COBOL functional program. Examples of these definitions are given in Figures 4.1 and 4.2, respectively. In order to facilitate the following discussion both figures have been assigned line numbers in such a manner that when the same area is defined it has the same line number. (This common line number assignment is also extended to Figure 4.3.)

It should be noted that Figure 4.1 was copied from a particular MCCRESSA program root and the values included are in some instances peculiar to that program. The tags are all padded to eight characters as is standard practice to improve the readability of the SETPRM keyword table located at the end of of the program root which typically includes references to some of these tags. (Note, instances of @ in the tags have been replaced with % to facilitate the preparation of this document.)

Alternatively, Figure 4.2 was copied from a COBOL copy code module named MFHDRPRM. This copy code module must be included in the LINKAGE SECTION of the COBOL program.

```
****************************** MASTER FILE HEADER PARAMETER - PART 1  01
         ALIGN                                                       02
MFHDRPRM EQU      *                                                  03
%NEWPAGE DATA     F'-2'          NEW PAGE CODE FOR "PRINTL" ROUTINE  11
%HDR1LNG DATA     F'132'         TEXT LENGTH FOR "PRINTL" ROUTINE    12
%%%%HDR1 DATA     CL132' '       HEADER 1 BUILT BY MFINFO            13
%BLINDSW DATA     CL001'Y'       Y -> NO DISPLAYS FROM MFINFO        31
%%MFTYPE DATA     CL001'E'       E -> EVAL MSTR, N -> NOMEN MSTR     32
#%%MFVOL DATA     CL002' '       MASTER FILE MCCRES VOLUME NUMBER    33
%%%%NAME DATA     CL010' '       MASTER FILE MCCRES VOLUME NAME      34
#MCATCNT DATA     CL001' '       NUMBER OF ACTIVE CATEGORYS          35
%MCATID1 DATA     CL002' '       CATEGORY CODE ID ONE                38
%MCATID2 DATA     CL002' '       CATEGORY CODE ID TWO                39
%MCATID3 DATA     CL002' '       CATEGORY CODE ID THREE              40
#MFEXCNT DATA     CL003' '       NUMBER OF EVALUATIONS               41
%%GROUPS DATA     CL160' '       MCCRES MASTER FILE GROUP CODES      42
%%%NOMEN EQU      %%GROUPS       NOMENCLATURE IF NOMEN FILE          43
****************************** MASTER FILE HEADER PARAMETER - PART 2  50
%%GCWRDM DATA     F'15'          GROUP CODE WORD MAXIMUM LENGTH      51
%%GCWRDL DATA     F'0'           GROUP CODE WORD CURRENT LENGTH      52
%GRPCWRD DATA     CL15' '        GROUP CODE WORD                     53
         DATA     C' '           NOT USED (ALIGNMENT ONLY)           60
****************************** END OF MASTER FILE HEADER PARAMETER   61
```

MFINFO MCCRES Header Information Parameter
EDL Definition

Figure 4.1

```
      *  STANDARD MCCRES HEADER INFORMATION PARAMETER          04
      *                                                        05
    0    2  FILLER              PIC S9999 COMP SYNC.          11
    2    2  FILLER              PIC S9999 COMP SYNC.          12
    4    2  MFHP-HDR1.                                        13
           3   MFHP-USERTL       PIC X(80).                   14
           3   MFHP-MSTRTL       PIC X(49).                   15
           3   MFHP-PNUM         PIC ZZ9.                     24
  136    2  MFHP-BLIND-SW        PIC X.                       31
  137    2  MFHP-TYPE            PIC X.                       32
  138    2  MFHP-VOL             PIC 9(2).                    33
  140    2  MFHP-NAME            PIC X(10).                   34
  150    2  MFHP-CATCNT          PIC 9.                       35
  151    2  MFHP-CATIDS.                                      36
           3    MFHP-CATID       PIC X(2) OCCURS 3 TIMES.     37
  157    2  MFHP-EXCNT           PIC 9(3).                    41
  160    2  MFHP-GROUPS.                                      42
           3    MFHP-NOMEN       PIC X(80).                   43
           3    FILLER           PIC X(80).                   44
  160    2  MFHP-GRPS-DEF2 REDEFINES MFHP-GROUPS.            45
           3    MFHP-GROUP       PIC X OCCURS 160 TIMES.      46
  320    2  FILLER              PIC S9999 COMP SYNC.          51
  322    2  FILLER              PIC S9999 COMP SYNC.          52
  324    2  MFHP-GC-WORD.                                     53
           3    MFHP-GC-CHAR     PIC X OCCURS 15 TIMES.       54
```

MCCRES Master File Header Information Parameter
COBOL Definition

Figure 4.2

4.1.1. Title Line

MFHDRPRM begins with a title line definition suitable for use as a parameter to PRINTL [6]. The first 80 characters (line 14) and the last three characters (line 24) of the title line (line 14) are not altered by MFINFO. Figure 4.3 gives the COBOL defined work area that MFINFO uses to initilize the MFHP-MSTRTL (line15). Here, HD-VOLNUM (line 17) will be set to the right justified Roman Numeral representation of the master file's MCCRES Volume number, HD-VOLID (line 19) will be filled in from the corresponding filed in master file, and HD-DATE (line 21) will be set to PARM-DATE (see Figure 3.1).

```
1   HDR-DETAIL.                                                  15
 2  FILLER          PIC X(03) VALUE SPACES.                      16
 2  HD-VOLNUM       PIC X(12) VALUE "VOLUME XXXXX".              17
 2  FILLER          PIC X(03) VALUE ":".                         18
 2  HD-VOLID        PIC X(10) VALUE "MISSING-ID".                19
 2  FILLER          PIC X(02) VALUE SPACES.                      20
 2  HD-DATE         PIC X(08) VALUE "MM/DD/YY".                  21
 2  FILLER          PIC X(06) VALUE SPACES.                      22
 2  FILLER          PIC X(05) VALUE "PAGE ".                     23
```

Header Detail Definition

Figure 4.3

4.1.2. Blind Switch

The Blind Switch, line 31, is used to control messages displayed from MFINFO. typically MFINFO displays one of the following two messages, as appropriate.

```
MASTER FILE IS VOLUME vv WITH eee EVALUATIONS.
NOMENCLATURE MASTER FILE IS VOLUME vv.
```

In the above, vv is the MCCRES Volume number and eee is the number of historical evaluations present in the file. To eliminate these messages, i.e., to suppress the appearance of the execution of MFINFO this switch should have a value of 'Y'.

4.1.3. Master File Type Switch

The master file type switch, line line 32, controls the type of MCCRES master file which may be processed by MFINFO. If this switch is initially blank, either type of MCCRES master file is acceptable and MFINFO moves the MCCRES file type indicator, E for evaluations and N for nomenclature, from the header record read into this switch. Otherwise, the initial contents of this switch must match the MCCRES file type indicator in the header record read, i.e., only that type file may be processed.

4.1.4. Remainder of Part One

The remaining fields of part one of the master file header parameter are filled directly from the header record read.

4.1.5. Part Two.

The group code word, line 53, is set only when an evaluations master file header record is read. In this case, each distinct group code appearing in %%GROUPS, line 42, is moved (in sequence) to the group code word and the length of group code word is set into line 52. The maximum length (line 51) will always be 15.

4.2. Program Messages

MFINFO issues informative and terminal error messages.

4.2.1. Informative Messages

Three informative messages may be issued by MFINFO. Two of these are controlled by the Blind Switch (see Section 4.1.2). The other message is controlled by the value of PARM-ID-SW (byte 24) of the standard control area. If this switch has a value of 'Y', MFINFO issues the following message.

GET MCCRES MASTER FILE HEADER INFORMATION - VERSION 2.0

4.2.2. Terminal Error Messages

MFINFO may issue one of eight terminal error messages by setting the return code (word one) of the standard control area and returning to the calling program root. The calling program root should then invoke PGMCTL to display the selected message(s) associated with that return code. Listed below are examples of card images which must appear in the program's PGMCTL control file for MFINFO terminal error messages.

4.2.2.1. ERR110 OPEN FILE ERROR \+4..

\+4. will have the file status item for the failed open request ([8] page G-1).

4.2.2.2. ERR120 READ FILE ERROR \+4. AFTER RECORD \5..

\+4. will have the file status item for the failed read request ([8] page G-2). \5. will have the value zero as only the header record is read by MFINFO.

4.2.2.3. ERR130 END-OF-FILE ERROR \+4..

\+4. will have the END-OF-FILE file status value of '10' which is illegal in MFINFO.

4.2.2.4. ERR140 HEADER RECORD NOT FOUND.

The record read did not have a level number of zero or a file type code of E or N. Therefore it is not a valid MCCRES master file header.

4.2.2.5. ERR150 MCCRES EVALUATION MASTER FILE REQUIRED -
ERR150 NOMENCLATURE MASTER FILE SUPPLIED.

4.2.2.6. ERR160 MCCRES NOMENCLATURE MASTER FILE REQUIRED -
ERR160 EVALUATION MASTER FILE SUPPLIED.

4.2.2.7. ERR170 CLOSE FILE ERROR \+4..

\+4. will have the file status item for the failed close request ([8] page G-3).

4.2.2.8. ERR180 ERROR CONVERTING VOLUME NUMBER TO ROMAN NUMERAL.

5. MCCRES Requirement Counts File Information (CRINFO)

CRINFO reads a MCCRES Requirement Counts File header record and maps it into a 2149 character parameter area in memory. Typically, this program is loaded by a MCCRESSA program root before loading the primary functional program. Its operation is similar to that of MFINFO.

CRINFO requires four parameters as follows:

a) Control Parameter,
b) Requirement Counts File Header Parameter,
c) Program Error Parameter, and
d) Program Identification Parameter.

The Control Parameter is the standard SETPRM control parameter which is normally copied into a MCCRESSA program using STDPARM copy code (see Figure 3.1). The Requirement Counts File Header Parameter is the object of this program. Finally, the Program Error and Program Identification parameters are required inter-program communication parameters for MCCRESSA programs utilizing PGMCTL (see Section 2.).

5.1. Requirement Counts File Header Parameter

Two definitions of the MCCRES requirement counts header paramater area are required to utilize CRINFO. These are the EDL definition which must be included in AN EDL root and the COBOL definition which must be included in a COBOL functional program. Examples of these definitions are given in Figures 5.1 and 5.2, respectively. In order to facilitate the following discussion both figures have been assigned line numbers in such a manner that when the same area is defined it has the same line number.

It should be noted that Figure 5.1 was copied from a particular MCCRESSA program root and the values included are in some instances peculiar to that program. Alternatively, Figure 5.2 was copied from a COBOL copy code module named CRHDRPRM. This copy code module must be included in the LINKAGE SECTION of the COBOL program.

PARM-PART1 (lines 12 through 18) are set directly from information contained in the header record read. Lines 21 and 22 are set to one more than the values contained in lines 14 and 15, respectively. Finally, lines 32 through 35 are set to information contained in the header record read as decoded by the subroutine IDDECODE [6].

```
********************************************** COUNT FILE HEADER PARM  01
         ALIGN                                                        03
         DATA      F'2'          LENGTH OF CFCATIDC                   04
CFPARM   EQU       *                                                  10
CFCATIDC DATA      CL002'MC'     CATEGORY IDENTIFICATION CODE         12
CFCATLEV DATA      CL001'1'      CATEGORY LEVEL                       13
CFOLDCNT DATA      CL003'000'    NUMBER OF COLUMNS IN INPUT RECORD    14
CFNEWCNT DATA      CL003'000'    NUMBER OF COLUMNS IN OUTPUT RECORD   15
CFGRPTAB DATA      CL015' '      GROUP CODES IN COUNTS FILE           16
CFVOLTAB DATA      CL024' '      VOLUME NUMBERS IN CNTS FILE          17
CFSECTAB DATA      CL015' '      SECTION CODES CNTS FILE              18
CFOLDNXT DATA      CL003'000'    OLD NEXT AVAIL COLUMN IN INPUT  RCD  21
CFNEWNXT DATA      CL003'000'    NEW NEXT AVAIL COLUMN IN OUTPUT RCD  22
CFCNTIDS DATA      CL2080' '     13X160 COUNT IDENTIFICATION WORDS    30
```

Requirement Counts File Header Information Parameter
EDL Definition

Figure 5.1

```
***** COUNT REQUIREMENTS FILE HEADER MEMORY FORMAT DEFINITION  01
* 2149 CHARACTERS                                    11/20/84   02
 1   CRFILE-HDR-PARM.                                           10
  2  PARM-PART1.                                                11
   3   CAT-ID              PIC X(2).                            12
   3   CAT-LEVEL           PIC 9.                               13
   3   OLD-CNT             PIC 9(3).                            14
   3   NEW-CNT             PIC 9(3).                            15
   3   IP-GROUP-TABLE      PIC X(15).                           16
   3   IP-VOL-TABLE        PIC X(24).                           17
   3   IP-SECT-TABLE       PIC X(15).                           18
  2  OLD-NXT               PIC 9(3).                            21
  2  NEW-NXT               PIC 9(3).                            22
  2  PARM-PART2.                                                30
   3   COUNT-ID            OCCURS 160 TIMES.                    31
    4     ID-VOL           PIC 9(2).                            32
    4     ID-EVAL          PIC 9(3).                            33
    4     ID-GROUP         PIC X.                               34
    4     ID-SECTS         PIC X(7).                            35
```

Requirement Counts File Header Information Parameter
COBOL Definition

Figure 5.2

5.2. Program Messages

CRINFO issues terminal error messages only by setting the return code (word one) of the standard control area and returning to the calling program root. The calling program root should then invoke PGMCTL to display the selected message(s) associated with that return code. Listed below are examples of card images which must appear in the program's PGMCTL control file for CRINFO terminal error messages. Note that these message numbers (return code) are the same as some of those issued by MFINFO. The user differentiates between the two programs by the issuing program identification that is also displayed when a terminal error message is displayed.

5.2.1. ERR110 OPEN FILE ERROR \+4..

\+4. will have the file status item for the failed open request ([8] page G-1).

5.2.2. ERR120 READ FILE ERROR \+4. AFTER RECORD \5..

\+4. will have the file status item for the failed read request ([8] page G-2). \5. will have the value zero as only the header record is read by CRINFO.

5.2.3. ERR130 END-OF-FILE ERROR \+4..

\+4. will have the END-OF-FILE file status value of '10' which is illegal in CRINFO.

5.2.4. ERR170 CLOSE FILE ERROR \+4..

\+4. will have the file status item for the failed close request ([8] page G-3).

6. Find Data Set (FINDDS)

FINDDS interacts with the user to identify an existing EDX data set and deliver to a prgoram parameter area certain EDX file characteristics. Typically, this program is loaded by a MCCRESSA program root before loading the primary functional program. It may be invoked as many times as necessary to identify all required files. Figure 6.1 gives the EDL definition of the format of the required parameter which is located indirectly, i.e., the program receives a one word parameter which is the address of such a 28 byte area.

```
*************************************************************************
PARMADR  DC      A(0)
PARM     EQU     *-*                                                (28)
PROMPT   EQU     PARM          ADDRESS OF PROMPT TEXT STATEMENT     (2)
DSN      EQU     PROMPT+2      MEMBER NAME  CHARACTER STRING        (8)
VOL      EQU     DSN+8         VOLUME LABEL CHARACTER STRING        (6)
RCDLNG   EQU     VOL+6         LOGICAL RECORD LENGTH                (2)
SIZE     EQU     RCDLNG+2      ALLOCATED EDX RECORDS                (4)
NEXT     EQU     SIZE+4        NEXT RCD DISPLACEMENT IN MEMBER      (4)
NLRCD    EQU     NEXT+4        NEXT RCD DISPLACEMENT IN BUFFER      (2)
*************************************************************************
```

FINDDS Indirect Parameter Area Map

Figure 6.1

6.1. Operation

PROMPT, its value represented below as pppppp, is used by FINDDS to construct two messages when interacting with the user. These are:

1) ENTER pppppp FILE NAME,VOL:
2) dddddddd,vvvvvv IS pppppp FILE?

where dddddddd,vvvvvv is the value of DSN and VOL, respectively.

The first message is issued a) upon entry if the initial value of DSN is blank, b) following a negative response to the second message, or c) following a positive response to the 'TRY AGAIN?' prompt which is issued after an open error condition. The second message is issued a) upon entry if the initial value of DSN is non-blank or b) following a user response to the first message.

The user may respond to the first message in one of three formats as depicted below.

```
dddddddd
dddddddd,vvvvvv
,vvvvvv
```

In the first format the user is supplying DSN, any existing value for VOL remains. It the second format the user is supplying both DSN and VOL. Finally, in the third format the user is supplying VOL, any existing value for DSN remains.

Following a positive response (Y) to the second prompt listed above, FINDDS attempts to open the subject data set. If the open is successful FINDDS then verifies that the data set is of the data type. If it is the fields RCDLNG, SIZE, NEXT, and NLRCD are filled in from the EDX DME for the subject data set and FINDDS terminates normally. Otherwise an error message is issued. All error messages are followed by the prompt 'TRY AGAIN?'. A negative response terminates the program abnormally with a return code of 16. In this case the returned value of DSN is blanks.

6.2. Error Messages

Error messages 6.2.1 through 6.2.3 are issued following a user response to prompt message (1), above. The remaining error messages are the result of a dificulty in opening the requested file. Each of these messages are preceded by the common open error message which follows.

```
pppppp FILE ON vvvvvv OPEN ERROR:
```

where pppppp is the value of PROMPT and vvvvvv is the current value of VOL.

6.2.1. NULL RESPONSE LEADS TO ABORT.

This message is issued when the user gave a null response to prompt message (1).

6.2.2. DATA MEMBER NAME INVALID.

The value supplied for DSN has a length of zero or greater than eight.

6.2.3. DATA VOLUME NAME INVALID.

The value supplied for VOL has a length of zero or greater than six.

6.2.4. dddddddd NOT FOUND.

The data set dddddddd was not found on volume identified in the preceding message.

6.2.5. VOLUME NOT INITIALIZED.

6.2.6. VOLUME NOT MOUNTED /UNUSABLE.

6.2.7. DIRECTORY I/O ERROR IN DSOPEN.

6.2.8. $$ DSN'S NOT ALLOWED.

6.2.9. dddddddd NOT DATA TYPE MEMBER.

The data set dddddddd is not flagged as data type in its DME.

REFERENCES

[1] BARZILY, Z. (1980). Analyzing MCCRES data. Technical Paper Serial T-427.

[2] BARZILY, Z., P. R. CATALOGNE, and W. H. MARLOW (1981). Assessing Marine Corps readiness. Defense Management Journal, Vol. 18, No. 1, pp. 25-29.

[3] BRIER, S. S., S. ZACKS, and W. H. MARLOW (1985a). An application of empirical Bayes techniques to the simultaneous estimation of many probabilities. Technical Paper T-486. To appear in Naval Research Logistics Quarterly.

[4] BRIER, S. S., S. ZACKS, and W. H. MARLOW (1985b). Results of a simulation study to measure the effectiveness of empirical Bayes' estimates of multiple probabilities. Technical Paper T-499 (forthcoming).

[5] CAVES, W. E. (1985a). Summary of the GWU Marine Corps Combat Readiness Evaluation System Software Applications (MCCRESSA). Technical Paper T-498.

[6] CAVES, W. E. (1985b). Marine Corps Combat Readiness Evaluation System Software Applications (MCCRESSA) subroutine library. Technical Paper T-501.

[7] CAVES, W. E. and W. H. MARLOW (1985). Marine Corps Combat Readiness Evaluation System (MCCRES) data base. Technical Paper T-503 (forthcoming).

[8] IBM CORPORATION (1980). IBM Series/1 Event Driven Executive COBOL Programmer's Guide. Document Number SL23-0014-1.

[9] ZACKS, S. and W. H. MARLOW (1982). Estimating the structural parameters of the Marine Corps Combat Readiness Evaluation System on the basis of the primary categories model. Technical Memorandum Serial TM-69200.

[10] ZACKS, S., W. H. MARLOW, and Z. BARZILY (1981). Category analysis of the Marine Corps Combat Readiness Evaluation System. Technical Paper T-450.

[11] ZACKS, S., W. H. MARLOW, and S. S. BRIER (1985). Statistical analysis of very high-dimensional data sets of hierarchically structured binary variables with missing data: an application to Marine Corps readiness evaluations. Naval Research Logistics Quarterly, Vol. 32, pp. 467-490.

END

9-87

DTIC